Построение простейшего синтаксического анализатора выражений

Теоретическая часть

Лексема - минимальная единица языка, имеющая самостоятельный смысл. Например, для языка C++ лексеммами являются ‘main’, ’38.9e-4’, ‘{‘ и т.п.

Синтаксический анализатор предназначен для распознавания синтаксической структуры из представленных лексических композиций (т.е. потока лексемм в том порядке, в каком они поступают на вход синтаксического анализатора). Проще говоря (опять в качестве примера возьмём C++), синтаксический анализатор должен «проглотить» последовательность лексемм **int a = 5;** и отказаться принимать такую последовательность тех же самых лексемм: **a = 5 int;**

Ниже будет рассмотрен один из множества алгоритмов синтаксического анализа арифметических выражений, позволяющий вычислить их значение.

Вычисление значений выражений с помощью обратной польской нотации

Из большого числа методов и алгоритмов вычисления значений арифметических выражений наибольшее распространение получил метод трансляции с помощью обратной польской записи, которую предложил польский математик Я. Лукашевич. Следуя высказыванию Ньютона, что "в изучении наук примеры важнее правил", рассмотрим этот метод на некотором обобщённом примере.

Пусть задано простое арифметическое выражение вида:

$$(A+B)*(C+D)-E \qquad (1)$$

Представим это выражение в виде дерева, в котором узлам соответствуют операции, а ветвям - операнды. Построение начнем с корня, в качестве которого выбирается операция, выполняющаяся последней. Левой ветви соответствует левый операнд операции, а правой ветви - правый. Дерево выражения (1) показано на рис. 1.

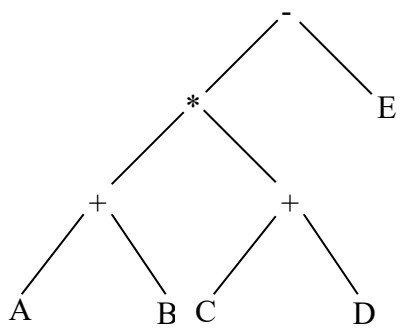

Рис. 1

Совершим обход дерева, под которым будем понимать формирование строки символов из символов узлов и ветвей дерева. Обход будем совершать от самой левой ветви вправо и узел переписывать в выходную строку только после рассмотрения всех его ветвей. Результат обхода дерева имеет вид:

$$AB+CD+*E- \qquad (2)$$

Характерные особенности выражения (2) состоят в следовании символов операций за символами операндов и в отсутствии скобок. Такая запись называется *обратной польской записью*.

Обратная польская запись обладает рядом замечательных свойств, которые превращают ее в идеальный промежуточный язык при трансляции. Во-первых, вычисление выражения, записанного в обратной польской записи, может проводиться путем однократного просмотра, что является весьма удобным при генерации объектного кода программ. Например, вычисление выражения (2) может быть проведено следующим образом:

# п/п	Анализируемая строка	Действие
0	A B + C D + * E -	r1=A+B
1	r1 C D + * E -	r2=C+D
2	r1 r2 * E -	r1=r1*r2
3	r1 E -	r1=r1-E
4	r1	Вычисление окончено

Здесь r1, r2 - вспомогательные переменные.

Во-вторых, получение обратной польской записи из исходного выражения может осуществляться весьма просто на основе простого алгоритма, предложенного Дейкстрой. Для этого вводится понятие стекового приоритета операций (табл. 1):

Операция	Приоритет
(	1
)	1
+ | -	2
* | /	3
^	4

Таблица 1

Просматривается исходная строка символов слева направо, операнды переписываются в выходную строку, а знаки операций заносятся в стек на основе следующих соображений:

a) если стек пуст, то операция из входной строки переписывается в стек;
b) операция выталкивает из стека все операции с большим или равным приоритетом в выходную строку;
c) если очередной символ из исходной строки есть открывающая скобка, то он проталкивается в стек;
d) закрывающая круглая скобка выталкивает все операции из стека до ближайшей открывающей скобки, сами скобки в выходную строку не переписываются, а уничтожают друг друга.

Процесс получения обратной польской записи выражения (1) схематично представлен на Рис. 2:

Просматриваемый	1	2	3	4	5	6	7	8	9	10	11	12	13	

символ														
Входная строка	(	A	+	B	)	*	(	C	+	D	)	-	E	
Состояние стека	(	(	+ (	+ (		*	(*	(*	+ (*	+ (*	*	-	-	
Выходная строка		A		B	+			C		D	+	*	E	-

Рис. 2

Задание

Написать калькулятор

Входные данные: произвольная строка символов.

Выходные данные: заключение о корректности входной строки как арифметического выражения, вычисленное значение этого выражения, в случае некорректной входной строки информацию о типе ошибки (например, «Несоответствие количества (и)»). Выходные данные зависят от сложности задания.

Синтаксический анализатор должен распознавать следующие лексеммы:

1. Знаки арифметических операций: ‘+’, ‘-‘, ‘*’, ‘/’
2. Скобки ‘(‘ и ‘)’
3. Действительные числа (т.е. дробные). Например: 12, 66.6, .54, 221.
4. Имя встроенной функции

Задание: считать строку символов, дать заключение о её соответствии арифметическому выражению, в случае несоответствия указать причину. Вычислить значение введённого выражения с учётом приоритета операций. При этом необходимо обрабатывать исключительные ситуации (например, деление на 0). Кроме того, нужно реализовать поддержку одной встроенной функции от двух переменных (например, log(a, b)). При этом аргументы функции сами являются выражениями (в том числе содержат эту функцию). Должна быть реализована возможность сравнительно простого изменения встроенной функции по требованию преподавателя (например, заменить log(a, b) на pow(a, b)). Встроенная функция имеет наивысший приоритет. Скобки служат для изменения порядка действий (обычная математика).

Синтаксический анализ и вычисление значения выражения могут быть раелизованы с помощью любого алгоритма (например, с помощью метода рекурсивного спуска и т.п.)